the AARDVARK JOURNAL

june 1981 vol 2, no.2

IN THIS ISSUE

Lots in issue. Am cramped for space. Complete Adventure, Article on how to write same. (1/2 article. Rest of magic stuff next month.) Articles on D & N Micro boards, Mittendorf High Res board, Final solution to Quick printer II problem, lots of letters, discussion of adding a disk.

Two nice programs by John Wilson. We just made available his BerZerKer program ($9.95) and he made us nice gift of two programs for Journal.

Pompea sent neat article to fix 65D file access. Got article on reverse video. This issue gonna cost AARDVARK a bundle in gift certificates.

Adventure Work Sheet

ROOM	N	S	E	W	U	D
1 RADIO ROOM	0	0	0	0	0	7
2 SHIPS BRIDGE	0	0	0	0	0	6
3 TOP OF FLAG POLE	0	0	0	0	0	5
4 OVER THE SIDE	0	0	0	0	6	0
5 ON THE BOW	0	6	0	0	0	0
6 MIDSHIPS DECK	5	7	0	0	2	12
7 AFT DECK	6	0	0	0	1	0
8 LIFE BOAT	7	0	0	0	0	0
9 ENGINE ROOM	10	0	0	0	0	-1
10 THE HOLD	0	-1	0	0	0	15
11 JANITOR'S CLOSET	0	0	0	12	0	0
12 HALLWAY	0	0	11	13	6	0
13 DOCTOR'S CABIN	0	0	12	0	0	0
14 CAPTAIN'S CABI	0	0	0	0	0	0
15 BILGE	0	0	16	0	10	0
16 ANCHOR CHAIN	0	0	0	0	5	0
17 BRIG	0	15	0	0	0	0
18 SWIMMING POOL	0	7	0	0	0	0

ADVENTURE WRITING

There is no kind of computing that I enjoy more than either playing or writing ADVENTURES. These come closest of any programs currently available to matching my pre-computing-days-mythological-picture of what a computer should do. When you are playing an ADVENTURE the darn machine seems to speak English. Instead of inputing "1" to go up and "2" to go down, you just tell the computer "GO UP" and it does it. It even talks back to you in English.

It also turns your little 8K desk top computer into a do-it-yourself build a world kit. The nice thing about the world that you build is that you can walk through it and pick up things, drop things, look at things and try things. You can build a place to fly spaceships, slay dragons, or climb mountains.

There are, of course, some slight problems inherent in getting a little bitty computer that speaks in numbers and has an I.Q. of effectively zero to do all of that. Take heart, however, while it took a long time to come up with techniques that will allow that to be done efficiently, it is like balancing an egg on end. It's only hard until you see how it's done.

To demonstrate the techniques, we're going to include a listing of DEATH SHIP in this JOURNAL. DEATH SHIP was the first ADVENTURE that I wrote and I make no pretentions that it is up to the standards we currently program at. It is a somewhat limited ADVENTURE a little less sophisticated than what we now do. It is however kind of fun and it does show all of the techniques that make a computer look so bright.

We are going to have a little problem discussing those techniques. It's one of those subjects where no matter what we cover first, there is something else that should have been covered before we could cover that, so bear with us a little. It may take a couple readings of this article to get it all straight.

The first thing I was concerned with in writing an ADVENTURE was to get the thing to speak English, or at least look like it spoke English. I did it by setting up two strings. If you look at the DEATH SHIP listing, line 290 gives you W$ which is the first two letters of each of the verbs that the system understands. Actually, W$ is GO + TAKE + DROP + TIE + REPLACE + OPEN + LOOK, and so on. We also have a string which contains the first two characters of each of the object words which the computer will understand. That's O$ which appears in line 700-710. The first twelve letters in O$ for instance stand for NORTH, SOUTH, EAST, WEST, UP and DOWN as those are six of the most common object words that the computer will have to cope with.

Now we have, in effect, two lists of words. One list of verbs and one list of all the objects. All we have to do is to input a string and figure out which of these words the player is using. We do put one constraint on the player. He has to use the normal English syntax for a command - verb first and then the object. In DEATH SHIP, we use the subroutine from 1730 to 1760 to input a string from the player and separate out the first two letters of the first word and the first two letters of the last word. We do that by looking for the last blank in the sentence being input and then taking the first two letters after the last blank (A$). For the first word we take the first two letters of the sentence as a whole.(B$)

The next important thing is to turn these groups of letters into numbers so we know what number verb and what number object we are talking about. We use the subroutine from 240 - 280 in DEATH SHIP to do that. All it does is to take A$ (the first two letters of the first word) and go through the string holding the first two letters of each of the verbs until it finds a match. When it finds one, it then picks up the number of the matched verb and goes to line 260 which does the same thing for the string holding the names of all the objects.(O$)

The only constraint on this system is that you are limited to having verbs and objects which do not share the first two letters of their names. For instance, you could not have a "CANDLE" and a "CAN" in the same adventure. While we used only two letters per word, you can decode more and get a wider range of vocabulary. However the more letters you decode, the slower the program will run.

At this point in the ADVENTURE, we have a number F which is the number of the verb and a number designated S which is the number of the object. Now we can start branching out and doing some things. The real secret to speed in the adventure is the ON-GOTO statement in line 300. Once we have F we can use the ON-GOTO to branch to individual subroutines to service each verb. F Line 300 contains one GOTO routine for each one of the verbs the system understands.

Now we have the English decoded into numbers, we need to take a step back and decide what we are going to decode. We have to design a universe and decide on some objects that we are going to put into it. We also have to design it in terms of being able to move around easily in it. To design the universe, the first thing we have to do is to draw a map. I won't include a map of DEATH SHIP here because of space limitations, but suffice it to say that when you go through it each location on the ship connects to each other location in one of six directions (NORTH, SOUTH, EAST, WEST, UP and DOWN). Among ADVENTURE programmers, each descreet location is usually referred to as a "room". That doesn't change the fact that it can be anything from a mountain top to a closet. I have included an example of an adventure worksheet such as we use during the writing operation. The first thing we do is to go through the map and name all locations and make up a numbered list of each location in the adventure. In DEATH SHIP there are a total of 18 descreet locations. Now we have to make a map of this whole thing that the computer can read. That's what the six columns on the right hand side of the worksheet are for. They are labeled the six directions (N, S, E, W, U, and D). What they contain is the number of the room that is in that direction from the room in that row. For instance, from the radio room which is location number 1, the first five directions are zeros because there are no rooms either north, south, east, west, or up, but the 6th element (down) is a 7 because room number 7 (the aft deck) is #7), you will note that it shows a 1 in the UP column because the radio room (#1) is "up" from the aft deck. It also shows a 6 in the first column because the midships deck is "north" from the aft deck. In that fashion, we go though the entire map and list in each column what room, if any, is in each direction from the room on that row. Once it is set up, it is easy to look at any location and see where you can go from there. For instance, look at the midship's deck entry (Row #6). From the midship's deck, room #5 is north, room #7 is south, room #2 is up room #12 is down and there are no rooms to the east and west. This can be read into the computer in an Nx6 array, N being the number of rooms, and 6 being the directions that can lead off that room. This is actually done in lines 760 - 800 in DEATH SHIP. It reads in six different numbers for each one of the rooms. We actually have a lot of blanks here because in a data statement the computer assumes a zero if there is nothing between the commas and in most instances there are only one or two exits from a room.

Now we can move the adventurer around his little universe. All we need to do is to set a single variable, in this case L, to be equal to the room that the character is in. Then when we decode his sentence, the first six objects are the directions. North is word #1, South is #2 and so on. All we have to do is to decode that row number for the room he is in (L) and see if there is anything in that object number to move to. For instance, if he is in room #2 and says 'GO DOWN' then S, the second word that we are going to decode is going to be a 6 and we look at P 2,6, find out that there is a 6 in that location and then we change his location to equal 6. He is now on the midship's deck. The lines that actually do that are from 890 through 1000. The line that does the actual change in position is line 1000. There are a lot of things in there that are logic constructs which we will discuss later, but the real meat of it is IFP(L,S) - that is the P value for the room where you are and the second word he put in. If it is greater than zero then the adventurer moves to that point in the array.

Now that we have all the descriptions listed by number and each one agrees with the number location you are in, all we have to do is to print D$(the room you are in) to picture where you are.

Before we can discuss how we can show the picture of things around you, we have to look at a second important array - that of all the objects in the game. There is a second worksheet that we use for that which is somewhat similar, but a lot simplier, than the map and direction worksheet. Each object has an associated name and number already and now has an associated location. We start by listing all of the objects on the sheet and numbering them. In DEATH SHIP and other ADVENTURES, I have used the convention of making all the portable objects contiguious in the lower half of the array and all the non-portable objects contiguious in the top half of the array. To see if an object is movable, I need only compare the number of the object with the number of the last movable object to see if it can be picked up and carried. Those things that have special conditions on being picked up and carried, as say water which would need a bucket, are put in between the portable and non-portable objects for slightly neater logic. We set up two arrays now, the first is O$(X) which has the name of each of the objects in the ADVENTURE. We also set up a location array L(X). In the L array we store the room number in which each of these objects will ultimately appear in. If they are to be off screen, say in a closed locker or not yet in existance until a player does something, then we give them a number higher than the last room number and they never appear in the descriptions. There is one complication in DEATH SHIP which I would probably change if I were to do it again. In an attempt to use all of the possible locations in the array, and knowing that N,S,E,W,U, and D would not need to have locations in that they were not that type of objects, I offset the locations six bytes from the object array. Therefore, L(X-6) equals the location of object X. It turned out that I had to type in so many X-6's that I didn't end up saving any space anyway.

So now we have the basic information we need to move around an ADVENTURE and look at everything. All we need to know is what room number the adventurer is in and we can print an associated description and then we scan through the array L(X) and anything that has the same number as that room number must be there and can therefore be printed along with the description. The line that allows us to do that are 1180 through 1210. Lines 1200 to 1210 tells you whether or not there are exits in the six directions by going through the row for the room number you are in and anything that has a number other than zero must be another room. I.E. if you are in Room 6, we word at P(6,1),P(6,12)..P(6,6) to see if anything is there. With just this much in the system we can walk around and see what is in every room. After this things get simplier - if a little bit more confusing.

Once you have this structure set up it is very easy to pick up and drop things and take them with you. We assigned one location, 0, to be the location number for the player himself. Anytime the machine scans through the L(X) array and find a zero that means that the player is carrying the item. Therefore, to have him pick it up all we need to do is to decode the word GET or TAKE look at the location of L(X) for that object's position and if it is the same as the room the player is in, L, then we set the location of that object equal to 0. If it's a viola, then he is carrying a viola. Of course we have to check to see if the object can be picked up. One mistake in an ADVENTURE lead to the embarassing ability of the adventurer to pick up a wandering stream and put it into his knapsack.

The routine to decode that in D.S. is in lines 360 to 510. The reason why the routine is so long when all it has to do is to find out if an object is next to you and make it equal to zero is that we put a lot of conditions on things that could be picked up or not picked up at certain times. Most of that is subsidary logic which we will discuss later in the game.

(TO BE CONTINUED IN AUGUST)

#	OBJECT	LOC	
1	NORTH	—	
2	SOUTH	—	
3	EAST	—	
4	WEST	—	
5	UP	—	
6	DOWN		
7	WIRE	50	L(1)
8	ROPE	3	L(2)
9	KEY	14	L(3)
10	AXE	17	L(4)
11	RADIO	1	L(5)
12	GLOVES	30	L(6)

```
 10 REM BASIC IN ROM USERS WHO HAVE NOT
REPLACED BASIC #3 WILL
 20 REM HAVE TO ADD THE SET UP TAPE FROM
 THE WORD PROCESSOR IN
 30 REM VOL1,#2 (THE STRING BUG FIX) AND
 ADD X=USR(X) TO LINES
 40 REM 250,260,720,1700,1720,1780
 100 FORX=1TO30:PRINT:NEXT:T1=5:T2=0:CA=
23
 110 DIMP(18,6),O$(44),L(44),D$(18):GOSU
B550:GOTO1010
 120 PRINT:PRINT"TELL ME WHAT TO DO":GOS
UB1730
 130 IFL(32)=L(18)ANDL(18)<>0THEN1620
 140 IFA$="HE"THEN1660
 150 IFF3=1THENFT=FT+1:IFFT=25ORFT=30THE
NPRINT"LIGHT GETTING DIM!!"
 160 IFFT>33THENF3=0
 170 T2=T2+1:IFT2=60THENT2=0:T1=T1+1:IFT
1=9THEN1520
 180 IFCA=24THENPRINT"IT'S TICKING LOUDE
R!":T3=T3+1:IFT3=14THEN1530
 190 IFF4>0THENPRINT"IT'S LEAKING!!":F4=
F4+1:IFF4=7THENF4=0:L(32)=18
 195 IFA$="CL"ORA$="JU"THENA$="GO"
 200 IFA$="GE"THENA$="TA"
 205 IFB$="RA"ANDL=6THENB$="ST"
 210 IFA$="SM"ORA$="BR"THENA$="HI"
 220 IFA$="TH"ORA$="PO"THENA$="DR"
 230 IFA$="IN"THENY=0:GOTO520
 240 FORY=1TOLEN(W$):IFMID$(W$,Y,2)=A$TH
ENF=(Y+1)/2:GOTO260
 250 NEXT
 260 FORX=1TOLEN(O$)STEP2:IFMID$(O$,X,2)
=B$THENS=(X+1)/2:GOTO280
 270 NEXT
 280 IFF1=1ANDF<>1THENPRINT"I DROWNED-I
TOLD YOU I CAN'T SWIM!!":END
 290 IFF<1THEN120
 300 ON(F-1)GOTO360,480,1440,1390,1350,1
240,1010,500,490,1560
 310 IFS=5ANDL(18)=0THENPRINT"CANT-CARRY
ING TO MUCH":GOTO120
 320 IFX>88ORY>22THENPRINT" HUH??":GOTO1
20
 330 F1=0:IFS=31ANDL(9)<>5THENPRINT"CANT
 YET-TO HIGH":GOTO120
 340 IFS=31ANDL(9)=5THENL=3:L2=3:GOTO116
0
 350 GOTO890
 360 IFS=19ANDL(14)=0THENF7=1
 370 IFB$="WA"THEN1580
 375 IFS=8THENO$(8)="ROPE":F2=0
 380 IFB$="HI"ANDL=10ANDL(17)=0THEN1510

 390 IFS>CATHENPRINT"I CANT":GOTO120
 400 IFS=19ANDL(14)<>0THENPRINT"NO HAMME
R":GOTO120
 410 IFS=11ANDF7=0THENPRINT"IT'S NAILED
DOWN":L(13)=1:GOTO120
 420 IFS=11ANDL(8)=50THENL(8)=1
 430 IFS<7THEN120
 440 IFS=8THENF2=0
 450 IFY>88ORL(S-6)<>LTHENPRINT"I DONT S
EE IT":GOTO120
 460 IFC=7THENPRINT"I CAN'T. I'M CARRYIN
G TO MUCH":GOTO120
 470 IFC<7ANDL(S-6)=LTHENL(S-6)=0:C=C+1

 480 IFF=3ANDS>6ANDL(S-6)=0THENL(S-6)=L:
C=C-1
 490 IFF=10ANDS=22THENF3=0
 500 IFF=9ANDS=22ANDL(16)=0THENF3=1:PRIN
T"OPPOSITE OF LIGHT=UNLIGHT"
 510 GOTO120
 520 PRINT"I'M CARRYING":FORX=1TO38:IFL(
X)=0THENPRINTO$(X+6):Y=Y+1
 530 NEXT:IFY=0THENPRINT"NOTHING
 540 GOTO120
 550 FORX=1TO44:READO$(X):NEXT
 560 DATA"NORTH","SOUTH","EAST","WEST","
UP","DOWN
 570 DATA"WIRE","ROPE","KEY","AXE","RADI
O","GLOVES","MESSAGE","CUTTER
 580 DATA"CRATE","PRESERVER","BUCKET","M
OP","NAILS","HAMMER"
 590 DATA"TIMEPIECE","FLASHLIGHT","SCREW
DRIVERS"
 600 DATA"BOMB WIRED TO A POST"
 610 DATA"LIFEBOAT","PORTHOLE","TABLE","
SWIMMING POOL"
 620 DATA"DOOR TO HOLD-LOCKED OTHER SIDE
","PORTHOLE","MAST","PORTHOLE"
 630 DATA"HEAVY HINGED LOCKED DOOR"
 640 DATA"ANCHOR CHAIN HANGING OVER THE
SIDE"
 650 DATA"EMERGENCY LOCKER","LOCKED DOOR
 - HAS WINDOW"
 660 DATA"SUPPLY CABINET","WATER","STARB
OARD RAILING","LOCKED BRIG"
```

```
 670 DATA"HINGES","A FAINT TICKING NOISE
","SIGN-ENGINE ROOM AFT"
 680 DATA"CLOSED DOOR"
 690 W$="GOTADRHITIREOPLOLIUNCU"
 700 O$="NOSOEAWEUPDOWIROKEAXRAGLMECUCRP
RBUMONAHATIFLSCBOLIPOT"
 710 O$=O$+"ASWDOPOMAPOHEANEMLOSUWASTLOH
IA SICL"
 720 FORX=1TO38:READL(X):NEXT:L=1
 730 DATA50,3,14,17,1,30,1,50,10,5,11,11
,50,10,2,29,31
 740 DATA9,7,14,14,7,12,4,5,16,10,5,8,12
,13,18,6,15,31,10,10,10
 750 FORY=1TO18:FORX=1TO6:READP(Y,X):NEX
TX,Y
 760 DATA,,,,,7,,,,,,6,,,,,,5
 770 DATA,,,6,,,6,,,,,5,7,,,2,12,6,,,,1
,
 780 DATA7,,,,,,10,,,,,-1,,-1,,,,15,,,,1
2,,
 790 DATA,,11,13,6,,,,12,,,,,,,,,,,,,16,,
10
 800 DATA,,,,,5,,,15,,,,,,,7,,,,
 810 FORX=1TO18:READD$(X):NEXT
 820 DATA"RADIO ROOM ","SHIPS BRIDGE"
 830 DATA"TOP OF FLAG POLE.","OVER THE S
IDE","ON THE BOW"
 840 DATA"MIDSHIPS DECK","AFT DECK","A L
IFE BOAT","ENGINE ROOM"
 850 DATA"THE HOLD","JANITORS CLOSET","A
HALL WAY","A DOCTORS CABIN"
 860 DATA"CAPTAINS CABIN","A DANK, SMELL
Y, WET BILGE"
 870 DATA"ON AN ANCHOR CHAIN","THE BRIG"
,"SWIMMING POOL"
 880 RETURN
 890 IF(B$="SW"ORB$="PO")ANDL=7THEN1540

 900 IFB$<>"PO"ANDB$<>"LI"THEN960
 910 IFL<>8ANDL<>7ANDL<>15ANDL<>16ANDL<>
4ANDL<>14THEN960
 920 IFL=4THENL=14:L2=L:GOTO1160
 930 IFL=14THENL=4:L2=4:GOTO1160
 940 IFL/2=INT(L/2)THENL=L-2
 950 L=L+1:L2=L:GOTO1160
 960 IFB$="AN"AND(L=5ORL=14)THENL=16:L2=
16:GOTO1160
 970 IF(B$="ST"ORB$="RA")AND(F2=0ORL(6)<
>0)THEN1370
 980 IF(B$="ST"ORB$="RA")ANDL=6THENL=4:L
2=4:GOTO1160
 990 L2=L:IFS>6THENPRINT"CANT GO THERE":
GOTO120
 1000 IFP(L,S)>0THENL=P(L,S):L2=L:GOTO11
60
 1010 IFB$="AR"ORB$=""THENL2=L:GOTO1160
 1020 IFB$="TI"ANDL(15)=0THENPRINT"TIME
IS "T1":"T2:GOTO120
 1030 IFLO=7ANDB$="LI"THENL2=8
 1040 IFB$<>"PO"THEN1090
 1050 IFL=7THENL2=18
 1060 IFL=16ANDF3=1THENL2=15
 1070 IFL=4THENL2=14
 1080 IFL=15THENL2=16
 1090 IFL=12ANDB$="WI"THENL2=14
 1100 IFL=15ANDB$="BR"THENL2=17
 1110 IFL=5ANDB$="MA"THENL2=3
 1120 IFL=6ANDB$="ST"THENL2=4
 1130 IFL=5ANDB$="AN"THENL2=16
 1140 IFL2<>LTHEN1180
 1150 PRINT"I SEE NOTHING SPECIAL":GOTO1
20
 1160 PRINT"I AM IN "D$(L):Y=0
 1170 IF(L=10ORL=15ORL=9)AND(F3=0ORL(16)
<>0)THEN1380
 1180 PRINT"I SEE ":FORX=1TO38:IFL(X)=L2
THENPRINT" *"O$(X+6);:Y=Y+1
 1190 NEXT:PRINT:L2=L:IFY=0THENPRINT"NOT
HING":PRINT
 1200 PRINT"OBVIOUS EXITS ARE":FORX=1TO6
:IFP(L,X)>0THENPRINTO$(X)
 1210 NEXT:GOTO120
 1220 L2=80:IFB$="WI"THENL2=14:IFL=14THE
NL2=12
 1230 GOTO1180
 1240 FORX=1TO44
 1250 IFL=8ANDB$="LO"ANDL(X)=29THENL(X)=
8:O$(35)="OPEN LOCKER
 1260 IFB$="CA"THENS=37
 1270 IFL=13ANDS=37ANDL(X)=30THENL(X)=13
:O$(37)="OPEN CAB.
 1280 NEXT
 1290 IFL=15ANDB$="BR"ANDL(3)=0THENP(15,
1)=17:O$(40)="OPEN BRIG"
 1300 IFL=14ANDB$="DO"THENP(14,2)=12:O$(
36)="OPEN DOOR":P(12,1)=14
 1310 IFL=10ANDB$="DO"THEN1330
 1320 GOTO120
 1330 P(10,1)=12:P(12,2)=10:O$(29)="OPEN
 DOOR":O$(44)=O$(29)
 1340 GOTO120
 1350 IFL(S-6)<>0THENPRINT"I DONT HAVE I
T":GOTO120
 1360 PRINT"ABANDONING SHIP-NO CHANCE OF
 FINDING BOMB BY 9:00":GOTO120
 1370 PRINT"MY HANDS SLIPPED! I AM DROWN
ING IN THE SEA":END
 1380 PRINT"IT'S DARK! I CAN'T SEE!":GOT
O120
 1390 IFS<>8ORL(2)<>0ORL<>6THENPRINT"CAN
T YET":GOTO120
 1400 INPUT"TO WHAT";A$:O$(8)="ROPE TIED
 TO "+A$
 1410 IFLEN(A$)>2THENA$=LEFT$(A$,2)
 1420 IF(A$="ST"ORA$="RA")ANDL=6ANDL(2)=
0THENF2=1
 1425 L(2)=L
 1430 GOTO120
 1440 PRINT"WITH WHAT":GOSUB1690
 1450 IFA$="FO"ORA$="FI"THENPRINT"YOU'RE
 KIDDING":GOTO120
 1460 IFL(4)<>0ORA$<>"AX"THEN1490
 1470 IFB$="DO"THENPRINT"STEEL DOOR-";
 1480 IFS=15ANDL(9)=LTHEN1500
 1490 PRINT"NOTHING HAPPENED":GOTO120
 1500 L(17)=L(9):O$(15)="SPLINTERS":GOTO
120
 1510 L(41)=0:O$(33)="DOOR LAYING ON FLO
OR":P(10,2)=9:GOTO120
 1520 PRINT"  THE TIME IS UP!!!":FORX=1T
O2000:NEXT:POKE56900,0
 1530 FORX=1TO20:PRINTTAB(X)"BBOOMMM!!!"
:NEXT:END
 1540 IFL(10)<>0THENPRINT"HURRY, I CANT
SWIM":F1=1
 1550 L=18:L2=18:GOTO1160
 1560 IFL(8)=0ANDB$="WI"ANDL=9THENCA=24:
L(1)=9:GOTO120
 1570 PRINT"NOTHING HAPPENED":GOTO120
 1580 PRINT"IN WHAT";:GOSUB1690
 1590 IFL<>18ORL(11)<>0ORA$<>"BU"THEN161
0
 1600 L(32)=0:F4=1:GOTO120
 1610 PRINT"I DONT THINK IT'L WORK":GOTO
120
 1620 PRINT"WHEW!!!!":PRINT"THE BOMB IS
DISARMED"
 1630 PRINTTAB(10)"THE TIME IS "T1":"T2
 1640 FORX=1TO5:PRINTTAB(5)"CONGRATULATI
ONS!":NEXT
```

```
 1650 PRINT"MAYBE NEXT TIME YOU WONT CUT
 IT SO CLOSE"
 1660 IFL=5THENPRINT"NOT ALL EXITS ARE O
BVIOUS"
 1670 IFL=1ANDL(8)=50THENPRINT"THERE IS
SOMETHING HERE YOU NEED"
 1675 IFL=6THENPRINT"LOOK AROUND CAREFUL
LY
 1677 IFL<>5ANDL<>6ANDL<>1THENPRINT"HOW
I SHOULD I KNOW-ITS YOUR ADVENTURE
 1680 GOTO120
 1690 INPUTA$:IFLEN(A$)>1THENA$=LEFT$(A$
,2)
 1700 RETURN
 1710 FORX=1TOLEN(A$):IFMID$(A$,X,1)ANDL
EN(A$)>X+1THENB$=MID$(A$,X+1,2)
 1720 NEXT
 1730 INPUTA$:FORX=1TOLEN(A$)
 1740 IFMID$(A$,X,1)=" "ANDLEN(A$)>X+1TH
ENB$=MID$(A$,X+1,2)
 1750 NEXT:IFLEN(A$)>1THENA$=LEFT$(A$,2)
 1760 RETURN
```

ADDING A DISK

One of the most frequent questions I have gotten over the last month or so has been "How Do I Add A Disk To My BASIC IN ROM System". There are a number of different options on how to do this.

There are some fixed requirements for adding a disk. Your system must first of all have 24K of RAM. OSI does sell a few systems with 20K of RAM but they do not really operate properly with OS65D until the extra 4K is installed. The second thing you've got to have is a Disk Drive. OSI uses the MPI Disk Drive and pretty well has us locked into using the one from the factory. The problem is that we need the Data Separator Card which is pretty much provided only with the OSI machines. Virtually, all of the 5-1/4" disk drives currently on the market are compatible, but very few of them have a data separator on board. The other thing we have to have is a Floppy Disk Controller Board.

There are a number of different ways to fill each one of these requirements. For instance, most of the OSI machines with BASIC IN ROM are capable of handling 8K of RAM on Board. That leaves us 16K of RAM short. There are 3 practical ways to get the extra 16K RAM. The first option is to buy the D & N Micro Products Memory Boards. That has the advantage of also filling the need for a Disk Controller. The second possibility is to purchase a Memory board from OHIO Scientific. Unfortunately, OSI has only been shipping the 24K RAM boards, and no longer offers the old 8 & 16K boards. That puts the price of the Memory up to about $475.00. It also puts you up to 32K RAM, but it is rather an expensive route to go. The third option is to buy either 2 of 8K or 1 of the 16K RAM boards from AARDVARK and populate it yourself. It's probably the cheapest way to get extra memory, but it does require that you do the building and troubleshooting and if you have a C4 or C8 system it does require that you wire up a long stringy connector to hook it up to your system. In any case, by one of those methods you need to get a 24K RAM in your system.

The other thing you need is a Floppy Disk controller. There are 3 now currently available and 1 or 2 more that should be available sometime in the next month or two. The old stand by is the OSI 470 board. It's OSI's original Floppy Disk Controller and has the same exact schematic as is currently used on the 505 and the C1P boards. It sells for about $130.00 populated and tested. It does have the disadvantage that it is set up for 8" disks when it's shipped from the factory and you, therefore will need to purchase a data sheet from AARDVARK to show you how to change a couple of resistors and some of the timing on it to run 5-1/4" disks. The second choice if you have a C2/4 system is to purchase the D & N Micro systems CM9 board either bare or populated. It has the disadvantage in that it is not assembled and tested with the interface unless you buy the full 24K system. It should come out to be fairly inexpensive if you build your own.

The other option is OSI's 610 board. The 610 will essentially hold 24K RAM and a Floppy Disk Interface. The board is designed for the C1P and comes with a Floppy Disk Interface assembled and tested and 8K RAM on board for $269.00. You have to add another 8k RAM before you can actually use the board. Fortunately, the sockets for the RAM are in so all you have to do is plug in more 2114's.

There are also two other Floppy Disk Controllers in proto right now which I would not wait for. Both AARDVARK here in the States and Looky Video in Australia have prototype wire wrap Floppy Disk Controllers running but neither firm has them in production and I wouldn't hold my breath for them.

None of these methods we have mentioned are limited to any one system. While the 610 board is designed primarily for the C1 it can be wired to the C2/4/8 backpane and some users have used it to upgrade C4 machines to Floppy Disk. The same thing goes for the D & N Micro board and the 470 board. While they are designed for the 48-pin buss, they can be wired through a cable to the 40-pin connector and they have been used by some people to add this to a C1P.

The third thing that you've got to have is a Disk Drive. There are three options open for the 5-1/4" disk drive. The Cleveland Computer Consumer Center in Cleveland Ohio, which appears to be a unofficial OSI factory store, sells the bare Disk Drive with cables but without case or power supply for $299.00. We've heard a number of rumors from OSI that that deal is about to end. If you want to get that one, you are going to have to hurry. The second possibility is to buy the Disk Drive from OSI. It comes complete with case, power supply and cables for $425.00. We are still checking out a third possibility. I recently found out that MPI sells Data Separators for their drives. According to MPI they sell them for $29.95. We've ordered one and should it work out as MPI promises, you can get any inexpensive MPI 5-1/4" disk drive and the data separater board from MPI and be in business. I hope it works out.

That then, I think, covers all the options on disk drives except OSI's. I should cover, I suppose, OSI's recommended procedure for taking a C24P and upgrading it to a C4 Mini Floppy. OSI recommends that you send the computer back to the factory where they remove the 502 CPU Card and replace it with a 505 CPU Card and a 527 memory board. That means, in effect that all they save from your old system is the case, power supply and 540 board. You also lose your cassette interface when you do that. I can think of no conversion I would recommend less than that one. It probably ranks right up there with the all time dumb ideas of the decade.

C1P SCROLLING INVERSE VIDEO HARDWARE MODIFICATION

BY LENNART LINDGARD, SWEDEN

Many times we OSI 600 board owners with envy have seen that more expensive personal computers such as PET, CBM and Apple have no difficulty what so ever making black characters on a white background on their monitors. This trick makes important parts of the information to step forward and increase in readability.

So, I asked myself, why not on my OSI? I looked at the schematics of the computer and saw that it would be rather simple to implement some sort of invertion of the video-signal out. But how was it to be controlled? My first thought was to cover every screen-position with an extra bit, which was to decide whether the sign on that particular position was to be inverted or not.

What I did, was to decode two different ASCII-characters, one to turn on "reverse video mode", and one to turn it off. Now I had to dig in the hardware.

The video unit on most microcomputers has a video-memory, that for each position on the screen contains the ASCII-value of the character that is to be drawn there. The video-memory sends out its ASCII-coded characters one at a time. Those characters are sent to a ROM; called "the character geneator". The character generator converts the ASCII-code to a signal which contains information about the physical looks of the character. That signal is transferred to a shift-register. The shift-register converts the parallel code into a serial one, which is sent out towards the video-monitor.

As I said earlier, I had to choose two characters out, which were to control the reverse video mode. I decided the signs, which disturbed the presentation as little as possible. That is, I chose the two SPACE's OSI provides in their character generator, character nr. 32 and character nr. 96. I chose to use nr. 96 to start the reverse video and nr. 32 to turn it off.

Binary 96 is written as: 01100000, and 32 is written as:00100000. As you can see, there is only one bit, bit 6, that differs those two binary numbers. That made it natural to trigg a "reverse

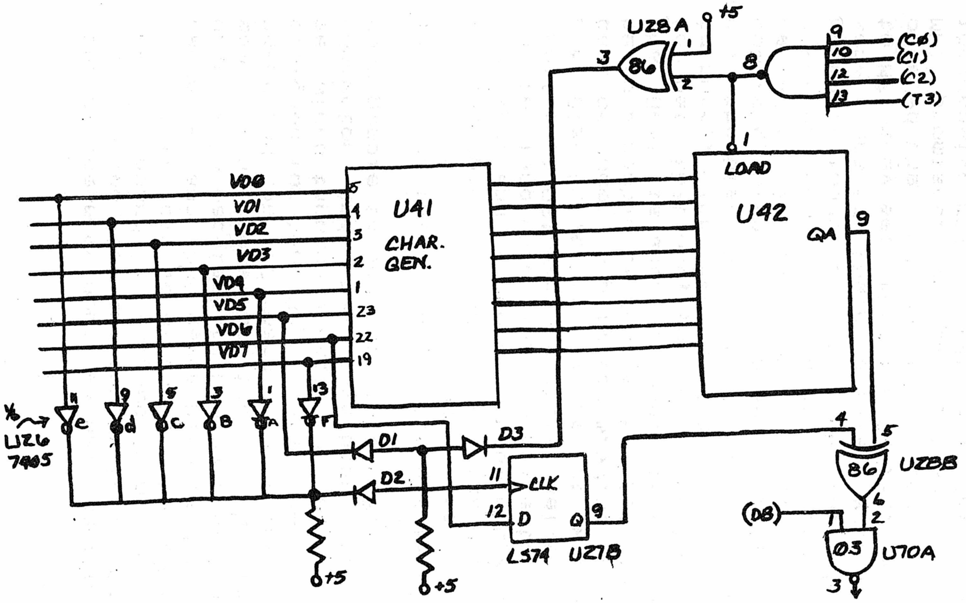


FIGURE 1

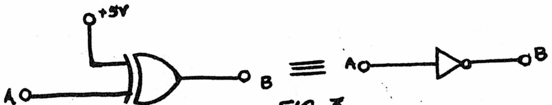


FIG. 3

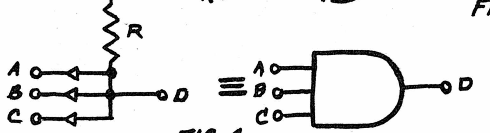


FIG 4

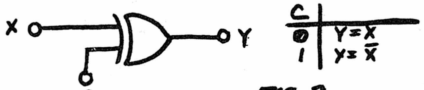
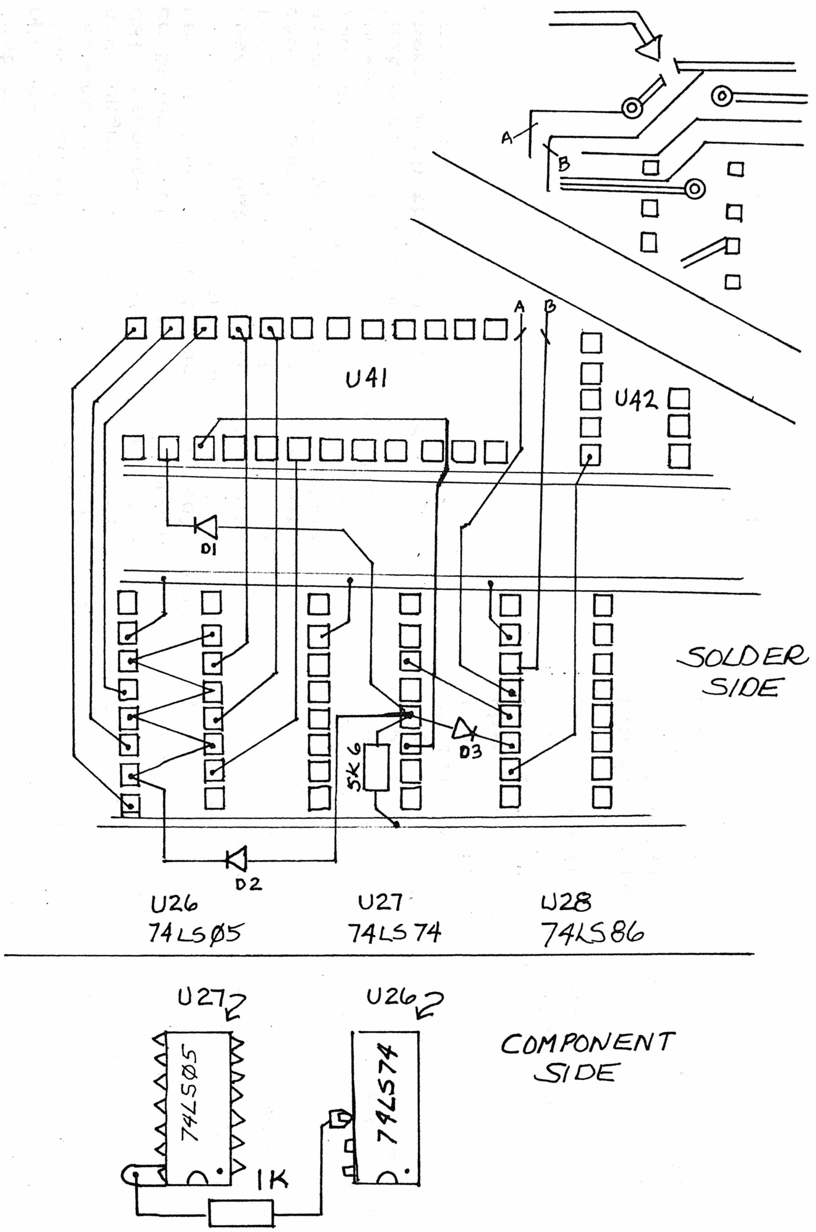


FIG. 2



video on/off" -flip-flop with the number: 0X100000 and use the X (the 6th bit) to decide the value of the flip-flop and the reverse video. On or Off.

To be able to control the video from the Q-output of the flip-flop, I had to cut the foil between pin 9 of the U42 (the shift-registers) and pin 2 of the U70, put in an XOR gate and control the XOR gate from the output of the flip-flop. The truth-table of the XOR gate is in fig. 2.

As the ASCII-code is transferred to the character generator (U41), long before the character-looks-signal is loaded in by the multiplexer for further delivery to the screen, the "load character-looks into multiplexer" -signal should also be added to the logic term, that triggers the flip-flop. That load-signal had to be inverted with a free XOR gate (see fig. 3) to fit.

All things needed to trigger the flip-flop, of course had to happen at the same time, that made it natural to use an AND-gate to collect all those things. I did use an AND-gate, but to save place on the board and to save IC's, I decided to use a diod-gate in discrete technique (see fig. 4). The gate should be built-up by germaniumdiodes because of their low voltage, 0.3 volts. Also note that I've used 7405-inverters instead of 7404's. That is because of the fact that the 7405 gates have an open-collector-output, which allows you to build an AND-gate by simply connecting the outputs of the gates together with one resistor connected to +5V.

Now you've got all the hardware information I've got to offer for this time. It's only for you to switch on your soldering iron, get tools, components and get started. But there is one thing, and you better read this carefully: when you start to modify your computer in any way, think of this:

All the warranties and insurances automatically cease to exist when you touch your computer with a soldering iron or a wire cutter. The company you bought the computer from has no reason for wanting to fix all the bugs you made when you fixed that great new hardware-mod.

If you are naturally handy with electronic things, this one shouldn't be very difficult. but don't say I didn't warn you.

Ok. That's about hardware. Now to the soft part of it all: The programming of your new reverse video modification.

When you're programming this mod, all you have to do to turn on the reverse video is to type "?CHR$(96);". The reverse video is automatically turned to normal again when it hits a SPACE. You can avoid the spaces by replacing them with dots:

```
Ex.    10  ?"NORMAL  VIDEO"CHR$(96)"INV.
VIDEO OFF"
```

You can also replace the spaces with "CHR$(96)";

```
Ex.   10 ?"NORMAL VIDEO"
CHR$(96)"INV"CHR$(96)"VIDEO OFF"
```

NEW BOARDS AVAILABLE FOR OSI

For several months now we've been hearing about a new company called D & N MICRO PRODUCTS, INC. They're located at 3684 N. Wells St. in Fort Wayne Indiana (Phone # 219-485-6414).

We've now talked to the D & N people, seen their catalog, and talked to a number of people who are using their products. The news has uniformally been good, and people who have gotten their products have been very pleased with them.

D & N Mico Products manufactures a number of boards for the OSI 48-pin buss. It means that all their boards are designed primarily to go on the C2/4/8 machines.

There are a number of boards which really wouldn't be of much use to a hobbiest or small business user, such as their Centronics Interface and RS232 boards. They do have a couple of boards, however, which are somewhat unique and which look to be extremely handy for the small user. The most interesting of these, would be what they call the BMEM-CM9 Memory Card. It's basically a 527 board replacement. It comes either bare as the BMEM card or with 8K, 16K or 24K of Memory on it. The prices range from $50.00 bare to $380.00 for a 24K board. The thing which makes it unique, however, is that it also contains a Floppy-Disk

Interface. It's the only way I know of currently to buy a bare memory board and Floppy-Disk Interface for OSI equipment. It comes populated with 24K in memory and the Floppy Disk Controller for $530.00.

We have had a number of customers now who have purchased these bare boards and they have had no particular trouble in getting them functional. D & N Micro recently sent us a sample copy of the instructions that come with the Mem 9 board and they appear to be extremely clear and easy to follow. You get a 15 page instruction manual including step-by-step instructions on how to restrap the ROMs and set the system up for either 5-1/4 or 8" Floppy's. D & N also makes a number of other boards and their catalog is certainly worth getting. They have, for instance, a prototype board for $33.00, OSI is supposed to have one of these, but I have yet to see them ever ship one. They also sell the 8-slot Backpanes, for the 48-pin buss, Parallel Printer Interfaces, Serial Interfaces, a number of different memory boards aside from the MEM-CM9, a Diablo Parallel Printer Interface and miscellaneous equipment.

In talking to the management at D & N, I found out that they have been in and out of the direct mail business over the last several months. They begain by advertising in a number of magazines and then found that the dealer business took up their entire production. They are now once again in the direct mail and retail business and seem to be planning on staying in that business now that their production can handle the increased demand.

Mr. Cass has upgraded the Word Processor from last year and added some features to take advantage of the capabilities of the C1E/C2E ROM. I added the routine at 1400 to allow editing of individual lines without retyping. That routine can be used with any ROM.

MINI WORD PROCESSOR

BY J.L. CASS, NORTHRIDGE CALIFORNIA

```
10 REM MINI WORD PROCESSOR
20 REM BY James L. Cass 1981
30 PRINT CHR$(26) TAB(18) "LINE
EDITOR":PRINT
40 PRINT"AFTER EXITING, RESTART PROGRAM
BY 'GOTO 100'
50 PRINT"OPTIONAL FOR PRINTing: M=LEFT
MARGIN, P=0 NO PAGING,
60 PRINT"P=1 FOR V (=54) LINE PAGES, P=2
NUMBERS PAGES.":PRINT
70 PRINT"USE NO QUOTES ("CHR$(34)") IN
COPY FOR TAPING
80 PRINT:PRINT"CALL COMMANDS WITH'/'; /H
OR /H FOR HELP.
90 PRINT:T=600:DIM
A$(T):L=1:V=54:M=0:B$=""
100 PRINT L;:INPUT A$:REM WARM START
WITH 'GOTO 100'
110 IF A$=""THEN A$(L)="":L=L+1:GOTO
100
120 IF ASC(A$)<>47 THEN
A$(L)=A$:L=L+1:GOTO 100
130 A=ASC(MID$(A$,2,1))
140 IF A=67 OR A= 99 THEN 310:CLEAR
SCREEN
150 IF A=72 OR A=104 THEN 400:HELP
160 IF A=86 OR A=118 THEN 500:VERIFY
170 IF A=70 OR A=102 THEN 520:FIND
180 IF A=65 OR A= 97 THEN 540:ADVANCE
190 IF A=68 OR A=100 THEN 600:DELETE
200 IF A=73 OR A=105 THEN 700:INSERT
210 IF A=85 OR A=117 THEN 760:UNSAVE
220 IF A=83 OR A=115 THEN 800:SAVE
230 IF A=82 OR A=114 THEN 850:REPLACE
240 IF A=76 OR A=108 THEN 900:LIST
250 IF A=78 OR A=110 THEN 980:NUMBERED
LIST
260 IF A=80 OR A=112 THEN 1000:PRINT
270 IF A=87 OR A=119 THEN 1200:WRITE
TAPE
280 IF A=84 OR A=116 THEN 1300:TAPE
INPUT
285 IFA=68ORA=100THEN1400:REM EDIT A
LINE
300 GOTO 100
310 PRINT CHR$(26):GOTO 100:CLEAR
400 PRINT"/LTR OR /LTR IS
COMMAND.":PRINT
410 PRINT"/H HELP /F FIND /A ADVANCE
420 PRINT"/L LIST /N NUMBERED LIST /P
PRINT
430 PRINT"/C CLEAR SCREEN /T TAPE INPUT
/W WRITE TAPE
440 PRINT"/I INSERT /D DELETE /R
REPLACE
450 PRINT"/S SAVE /U UNSAVE /V VERIFY
460 PRINT"/E EDIT A LINE":PRINT
470 PRINT"START WITH QUOTE (CHR$(34)")
IF LINE STARTS
480 PRINT"WITH SPACE OR CONTAINS ANY
COMMAS OR COLONS.
490 PRINT:GOTO 100
500 PRINT Q; A$(Q):GOTO 100:VERIFY
520 INPUT"#";Q:GOTO 500:FIND
540 INPUT"#";R:IF Q+R>T THEN 100
550 IF Q+R<1 THEN Q=1:GOTO 500
560 Q=Q+R:GOTO 500
600 INPUT"DELETE";X:IF X>L THEN 100
610 PRINT A$(X):INPUT"O.K.";A$:IF
ASC(A$)<>89 THEN 100
620 L=L-1:FOR Y=X TO
L:A$(Y)=A$(Y+1):NEXT:GOTO 100
700 F=0:REM INSERT
710 INPUT"AFTER LINE";X:IF X>L THEN 100
720 IF F=0 THEN INPUT A$:IF A$="NO" THEN
100
730 L=L+1:FOR Y=L TO X+2
STEP-1:A$(Y)=A$(Y-1):NEXT
740 A$(X+1)=A$:GOTO 100
760 F=1:A$=B$:GOTO 710:REM UNSAVE
```

```
800 INPUT"SAVE LINE";R:IF R>T THEN 100
810 B$=A$(R):GOTO 100
850  INPUT"REPLACE  LINE";R:IF  R>T THEN
100
860 PRINT A$(R):INPUT A$
870 A$(R)=A$:GOTO 100
900 F=0:REM LIST
910  INPUT"START LINE (A  FOR
ALL)";A$:A=ASC(A$)
920 IF A=65 OR A=97 THEN X=1:Y=L-1:PRINT
CHR$(26):GOTO 940
930 X=VAL(A$):INPUT"TO LINE";Y
940 FOR I=X TO Y:IF F=1 THEN PRINT I;
950 PRINT A$(I):NEXT:PRINT:GOTO 100
980 F=1:GOTO 910:NUMBERED LIST
1000 PRINT CHR$(26):REM PRINT
1010 POKE 517,1
1020  FOR I=1 TO  6:PRINT:IF I=2 AND P=2
THEN PRINT"PAGE 1":I=3
1030  NEXT:R=2:FOR I=1  TO  L-1:PRINT
SPC(M+1) A$(I)
1040 IF I/V<>INT(I/V) OR P=0 THEN 1090
1050 FOR X=1 TO 7:PRINT:NEXT
1060 IF P=1 THEN PRINT
1070 IF P=2 THEN PRINT"PAGE"R
1080 R=R+1:FOR X=1 TO 4:PRINT:NEXT
1090  NEXT:FOR X=1 TO 8:PRINT:NEXT:POKE
517,0:GOTO 100
1200  PRINT"START  TAPE  AND HIT <SHIFT>
WHEN READY.
1210 POKE 57088,1:WAIT 57088,6
1220 POKE 517,1:FOR X=1 TO L-1
1230 PRINT CHR$(34) A$(X)
1240 NEXT:PRINT"**":POKE 517,0:GOTO 100
1300 PRINT"START TAPE  AND HIT <SHIFT>
WHEN READY.
1310 POKE 57088,1:WAIT 57088,6
1320 POKE 515,255
1330 INPUT A$:IF A$="**" THEN 1350
1340 A$(L)=A$:L=L+1:GOTO 1330
1350 POKE 515,0:GOTO 100
1400INPUT"EDIT  WHAT
LINE";E:IFE>LTHEN100
1410?A$(E):INPUT"CHANGE";C$:IFC$="N"THEN100

1420INPUT"TO
";NW$:A$=A$(E):FORY=1TOLEN(A$)-LEN(C$)+1

1430IFMID$(A$,Y,LEN(C$))=C$THEN1450
1440NEXTY:?"NOT FOUND":GOTO100
1450A$(E)=MID$(A$,1,Y-1)+NW$+ MID$
(A$,Y+LEN(C$)) : GOTO100
1999 END
```

IMPROVING RANDOM ACCESS FILES ON 65D

BY DAVE POMPEA

I have two programs, the first is a patch for disk random files (LUN #6), the second program will put a volumn # and title on your disks (hide it in the unused space at the end of sector #4 on track 12)

There is a flaw in the code that OSI uses to access random files. Every time you do a GET,N OS65D calculates the track to find that record on and then proceeds to read that track, even if that save track is already in memory. Remember that using 128 byte records, 16 records will fit on each track. That means that if your program is searching the file for a record it will be verrrry slow. The patch fixes this. If the track is already in memory, it skips the disk read.

To use the patch program, include it at the start of your program and put your file name in the OPEN stmt. The patch grabs some memory at the top of RAM, POKEs in the new code, and POKEs in a jump to it after the OPEN statement.

Also, the commands OPEN, CLOSE, GET & PUT don't have to be spelled in full, all you need is the first letter, DOS skips the rest up to the comma.

"A PROGRAM TO CREATE VOL# & NAME ON DISKS"

```
5 REM PGM NAME IS VOL.CR
7 REM by Dave Pompea for Aardvark
Journal
100 REM CREATES VOLUMN AND NAME OF DISK
120 REM
130 PRINT:PRINT:PRINT"VOLUMN & DISK
NAME":PRINT
140 INPUT"ENTER VOLUMN # (1-255) ";VN
150 PRINT:PRINT"ENTER NAME OF DISK"
160 INPUT DN$
170  IFLEN(DN$)>15THEN PRINT" *MUST BE
LESS THAN 16":PRINT:GOTO150
180 DN$=CHR$(VN)+DN$
190  IFLEN(DN$)>16  THEN DN$=DN$+"
":GOTO190
195 REM Get overlay #4
200 DISK"CA 2E79=12,4"
220 FORI=1TO16
225 REM POKE Vol # & Name to free space
230 POKE12135+I, ASC ( MID$ (DN$,I,1))
240 NEXT
245 REM Put back on Disk
250 DISK!"SA 12,4=2E79/1
260 PRINT:PRINT"*EOJ VTOC": PRINT
270 END
```

ADD THESE LINES TO YOUR DIRECTORY PRINT PGM. (DIR OR BEXEC*)

```
140 DV=2:PRINT:PRINT
141 DISK!"CA 2E79=12,4
142 VN=PEEK(12136): IFVN=0THEN150
143 PRINT:PRINT"VOL.#";VN;
144 FORI=12137TO12152 :
PRINTCHR$(PEEK(I));: NEXT: PRINT: PRINT
150 REM GOTO  HERE IF NO VOL # & NAME ON
DISK
```

```
  10   ; FIX FOR RANDOM FILES (#6)
  20   ;
  30   ; BY D. POMPEA
  40   ;
  50   ; COPYRIGHT 1981
  60   ;
 100 8000          *      = $8000
 120 8000 AD2C23          LDA $232C ;TRACK
 IN MEMORY
 130 8003 8D1780          STA TR.MEM
 140  8006 20CA2E          JSR $2ECA ;FIND
 TRACK BY RECORD #
 150 8009 AD1780          LDA TR.MEM
 160 800C CD2D23            CMP $232C ;NEW
 TRACK
 170 800F F003             BEQ SKIP  ;SAME
 AS OLD
 180  8011 206729          JSR $2967 ;DISK
 READ
 190 8014 4C8E22 SKIP        JMP $228E
 ;RETURN
 200 8017 00      TR.MEM..BYTE 0
 .!IO 02,02
```

```
 10 REM PGM NAME IS RB.FIX
 20 REM
 30 REM BY DAVE POMPEA
 40 REM
 50 REM PATCH  SO THAT GETS FROM FILE #6
DON'T READ DISK EVERY TIME.
 60 REM ONLY WHEN NEW TRACK
 70 REM
 90  REM GET SPACE AT TOP OF MEMORY  FOR
PATCH
100 BA=PEEK(133)
110 POKE133,BA-1
120 REM GET CODE & POKE IT IN
125 FORI=0TO23
130 READC:IFC=31THENC=BA
135 POKEBA*256+I,C
140 NEXT:CLEAR
200 DISKOPEN,6,"Your File name herA
135 POKEBA*256+I,C
140 NEXT:CLEAR
200 DISKOPEN,6,"Your File name here"
210 REM OPEN GETS SECTOR 4 FROM TRACK 12
& STORES AT $2E79 (DIR BUFFER
220 REM
230 REM GET CODE ADDRESS (BA ZEROED WITH
'CLEAR')
240 BA=PEEK(133)+1
245 REM POKE IN JMP TO PATCH
250 POKE11966,76
260 POKE11967,0
270 POKE11968,BA
300  DATA173,44,35,141,23  ,31,32,202,
46,173,23,31,205,44
310 DATA35,240,3,32,103, 41,76,142,34,0
```

MEMORY RELOCATOR
BY CHARLES A. STEWART

Since I purchased the EPROM Burner from AARDVARK I have been toying with the idea of placing many routines I utilize on EPROM, because with a cassette based system the OSI ASSEMBLER EDITOR would be very convient to use if it could be booted in from PROM instead of the 8 minute load from tape.

The major problem with these machine language programs is they start load in page 2 and in the case of the assembler run thru $1396. If the program resides in the norm basic workspace as the assembler and AARDVARK CHESS PROGRAM do the task connot be done in Basic. After some trial and error I developed a short program which will move any program - MACHINE LANGUAGE OR EVEN BASIC to any location in RAM in a twinkling of an eye. The routine is short and simple enough to enter into the monitor when you require it buy may of course be saved on tape.

Listing 1 is a simple Basic program which may be used to relocate programs into RAM from addresses above page 3 and of course Basic and Monitor Roms. The routine is very slow in comparison to the equivilent machine language routine. I utilize a move from Monitor Rom to $2000 in this example.

LISTING 1

```
  10 Z=0:FORX=63488 TO 65535  :
Y=PEEK(X)
  20 POKEZ+8192,Y : Z=Z+1 : NEXT
```

Machine language routines - May be used for any memory locations and is relocatable by changing the incrument page address to correspond to the proper location.

"LISTING 2"

```
10  0000            ;MEMORY RELOCATOR VER
1.2
20 0000           ;BY CHARLES A. STEWART
60 0000           ;
70 0000           MON=$FE0C
80 0000           FROM=$1F06
90 0000           TO=$1F09
100 0000          ;
110 1F00          *=$1F00
120  1F00 A008      LDY  #$08 ;LOAD WITH
NUMBER OF PAGES TO RELOCATE 8=2K
130 1F02 A200     LOOP1 LDX #$00
140  1F04  BD00F8     LOOP2 LDA $F800,X
;START ADDRESS OF PROG TO RELOCATE
```

```
150 1F07 900020  STA  $2000,X ;LOCATION
TO PUT TO
160 1F0A E8       INX
170 1F0B D0F7     BNE LOOP2
180 1F0D EE061F   INC FROM
190 1F10 EE091F   INC TO
200 1F13 88       DEY
210 1F14 D0EC     BNE LOOP1
220 1F16 4C0CFE   JMP MON .A2
```

The machine language equivalent seems longer and more complicated but just try running both. The Basic PEEK and POKE routine requires 27 sec to run, the machine language equivalent runs so fast that you might not even know it ran so I jumped to the monitor at $FE0C to indicate that the routine has completed its task, run time is under 1 second!

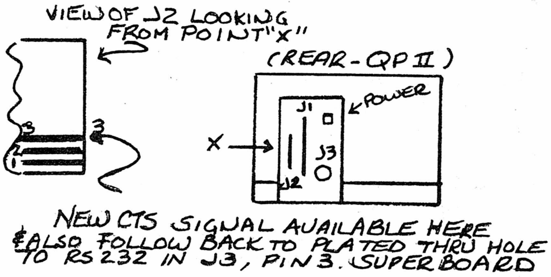
I have utilized this method to relocate the OSI ASSEMBLER/EDITOR and AARDVARK CHESS PROGRAM and burned to EPROM. I have also disassembled them both to learn how they operate.

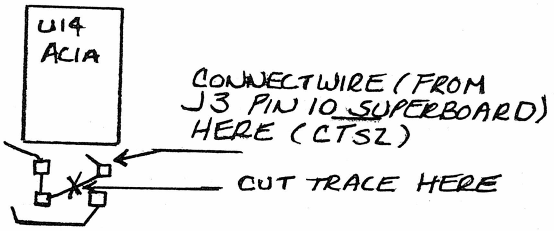
```
10  0000             ;PROGRAM TO READ FROM
EPROM
20 0000             ;BY CHARLES A. STEWART
60 0000             ;
70 0000             ;
80 1F30             *=*1F30
90 1F30 A200        CONT LDX #$00
100 1F32 A900    HERE LDA #$00
110 1F34 8D02F2 STA $F202
120 1F37 A900    THIS LDA #$00
130 1F39 C908    CMP #$08
140 1F3B D003    BNE HERE2
150 1F3D 4C5A1F JMP END
160 1F40 8D01F2 HERE2 STA $F201
170 1F43 AD0F32 LDA $F203
180  1F46  9D0000  STORE  STA
$0000,X;ADDRESS TO PUT TO
190 1F49 EE331F INC HERE+1
200 1F4C E8       INX
210 1F4D D0E3    BNE HERE
220 1F4F EE481F INC STORE+2
230 1F52 EE381F INC THIS+1
240 1F55 A900    LDA #$00
250 1F57 4C301F JMP CONT
260 1F5A 4C0CFE END JMP $FE0C
```

The article below should be the final word (I hope) on how to interface the Quick Printer II. - No more software patches - No more weird POKEs.

MR. R. WRAIGHT, HANTS, ENGLAND

The enclosed hardware modification will mate a Quick Printer II on a Superboard without any software patch. As mentioned in previous Journals POKEing in loads of nulls etc. works, but as the printer does not like nulls you sometimes get spurious characters printed at the beginning of the line. The Mod is as follows:-on the back of the printer there are two edge connectors intended for Radio Schack computers on one of these there is a signal called "Line". This signal is normally high, but goes low everytime print head is to move or moving. We can use this to control the C.T.S. line on the ACIA assuming you've already fully populated your RS232 IN and OUT. The new C.T.S. line from the printer should goto J3 pin 3 (RS232 in). The CTS2 can now be picked up at J3 pin 10 (CTS2). We now only need to link CTS2 to the ACIA pin 24. This is easily accomplished as then a line pad (W3) just by the ACIA. Cut the track as shown on sketch and connect CTS2 to pad shown. This completes the Mod to the Superboard. For the more ambitious it is possible (two of the screws are under the rubber feet) watch out for the led. The new CTS can be wired to the Din plug internally by tracing the shown edge connector track back to a plated thr'o hole and soldering a wire to this and then take other end of wire to an unused pin of the Din plug. Then all connections from printer are still contained in one cable. The only restriction to above software is that the terminal width should be restricted to 31 i.e. POKE 15,31. The reason for using the RS232 in for CTS is to ensure that no spurious voltages etc. can get into Superboard from the printer. I hope the above information is of some use to others.


VIEW OF J2 LOOKING FROM POINT "X"
(REAR-QP II)
POWER
J1
J3
J2
X
3
3
2
1
NEW CTS SIGNAL AVAILABLE HERE
&ALSO FOLLOW BACK TO PLATED THRU HOLE TO RS232 IN J3, PIN 3. SUPERBOARD



LETTERS, LETTERS, AND MORE LETTERS

LEON DAUGHERTY, RANCHO PALOS VERDES, CALIFORNIA

I wrote you last week about my disappointment that the CE2 ROM would not load WP6502. You might be interested in how I have temporarily solved this problem. You might want to include the following as a suggested alternative way of installing the CEGMON for anyone who is using WP6502.

Follow the printed instructions except for the following. After removing the three jumpers at U15 install a 14 pin DIP socket at U15. Connect the wires from the 74154 pins 9 thru 11,13, and 15 thru 17 to a 14 pin DIP plug in accordance with the sequence in the printed instructions. The wires should be long enough to allow easy insertion and removal of the plug in the socket at U15. On a second 14 pin DIP plug connect jumpers from pin 1 to pin 12, pin 2 to pin 11, and pin 3 to pin 10.

Remove the jumpers or cut the foil at W4 and W5. Cut two 2" lengths of wire. Strip one end of each 1/8" and the other end 1/4". Using the 1/8" stripped ends solder a wire to the center hole at both W4 and W5. I used a #58 drill to clean the solder out. Obtain a screw terminal tip for printed circuit for printed circuit boards with a .200" center spacing. I used and OK Machine and Tool, No. TS-8. I cut thru the terminal strip at terminals 3 and 6. This provided two 2 screw terminal strips with approximately the spacing of the jumper holes at W4 and W5. With slight bending of the solder pins to fit the spacing of the holes you can solder one of these strips at W4 and the other at W5. When soldering these be sure the solder flows on both sides of the PCB.

Remove the Monitor ROM and set it aside. Carefully unsolder and remove the monitor socket U5. Install a Zero Insertion Force Socket at U5. I found it necessary to move one bypass capacitor to the back side of the board to provide room for this socket. One other had to be moved and soldered to the ground foil instead of the hole it was in.

This completes the changes in the installation proceedure. You now have a 502 board that can be easily converted from the SYNMON to CEGMON and back as needed. For SYNMON insert the wires at W4 and W5 in the leftmost terminals and tighten the screws. Insert the plug with the jumpers at U15 and install the SYNMON ROM at U5. You are now in business for using WP6502.

For CEGMON move the wires at W4 and W5 to the right side, remove the plug with the jumpers from U15 and insert the plug with the wires from the 74154. Remove the SYNMON and install CEGMON. All the conveniences of CEGMON are now available.

This is an inconvenient changeover because it does require removing the bottom cover each time, but since there is no soldering and the Zero Insertion Force Socket makes changing the ROM easy and safe, it only takes a few minutes to make the change.

JAMES C. SCHMOOCK, SAN JOSE, CALIFORNIA

Here is line which was missing from the SLASHBALL program in the April 1981 JOURNAL. This line allows the program to run on the C1. Without it the ball gets served off the screen.

```
   166  IFVB=600THENPN=
INT(54148+24*RND(8)) :MF=L
```

The JOURNAL is great - keep it up. Request - how about some discussion of how to make Machine Code programs run with the C1S ROM. Specifically how to use info in the C1S instructions like getting "E-Z LISTER" and other published programs to run.

LESTER CAIN,

Did you know it is possible to load Assembly source code into BASIC workspace and vice versa? By using the Indirect File function one can dump either Source into the others work area. One possible use for Basic in Assembly would be to delete blocks of lines, but care must be taken as long lines will be shortened.

By putting Assembly source into Basic a person can edit the file with Aardvarks' Editor or renumber it with different values other than what is available with the Assembler. There is a hazard here also in that the first spaces are removed and won't look as neat.

While on the subject, can anyone tell me where the values for the Assembler's resequence are located? It is a hassle to use line 10 as a starting place all the time. I have written my own resequence routine to do this but then it has to be assembled also and takes more time. Any information on this would be appreciated.

PETER BUTTERFIELD, PALO ALTO, CALIFORNIA

Thank you for your letter of April 27th, responding to my questions about the assembler on the OSI game disk. Your statement that it was for a C4P provided the info I needed to make it work. I simply patched the Assembler's keyboard routine as shown below, to complement the characters stored and read at DF00, and now it works fine! Thank you!

Regarding reading disks at 2MHZ: When I made the change, I tapped the clocking signal at the next lower frequency (the clocking signal for the disk circuit, that is), so that the data rate to and from the disk remained as before. So far it has worked with no problems.

```
:Q1547

ASSEMBLER AS SUPPLIED ON GAME DISK

1547 AD00FC   LDA $FC00
154A 4A       LSR A
154B 900D     BCC $155A
145D 200FF24  JSR $24FF
1550 4903     EOR #$03
1552 D0ED     BNE $1541
1554 8D2523   STA $2325
1557 4C0002   JMP $0200
155A 206015   JSR $1560
155D 4C5015   JMP $1550
1560 A9E1     LDA #$E1
1562 206D15   JSR $156D
1565 5005     BVC $156C
1567 4A       LSR A
1568 4A       LSR A
1569 4A       LSR A
156A 90F6     BCC $1562
156C 60       RTS
156D 8D00DF   STA $DF00
1570 2C00DF   BIT $DF00
1573 60       RTS
1574 00       BRK
1575 00       BRK
  (00 THROUGH 16FF)

 CORRECTIONS FOR C1PMF:

1547 AD       LDA $F000

156D          JMP $1575
1570 2C       BIT $1574
1573 60       RTS
1574 00       BRK
1575          PHA
1576          EOR #FF
1578          STA $DF00
157B          LDA $DF00
157E          EOR #FF
1580          STA $1574
1583          PLA
1584          JMP $1570
```

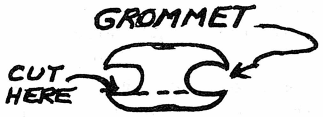
DONALD R. BRENNAN, NORTHBORO MASSACHUSETTES

"10 CENT BREAK KEY MOD"

This mod will eliminate accidental Break Key operation. A 3/8" ID X 1/4" H rubber grommet is used to insure a moderate pressure is required to operate the Break Key.

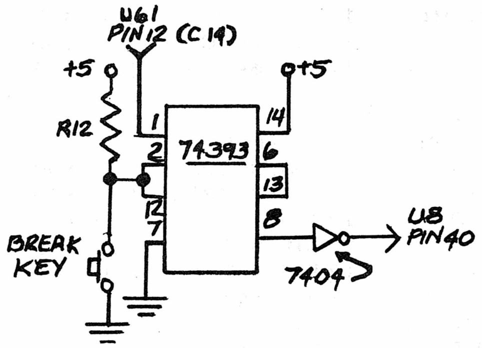
Gently pry up the Break Key Cap using your fingers. Then cut off the lower portion of the grommet as shown.

Place the top portion of the grommet over the Break Key Switch Top with the large end down. Now gently install the Break Key Cap. Power up and notice Break Key must be pushed firmly to operate and no more accidental Break Key operations will occur.



CHUCK SCHALL, WEST HAVEN CONNECTICUT

After reading in the April AARDVARK Journal that the new Superboard provided a three second delay on reset, I decided to try and implement a similar modification to my C1P. The easiest method that I found was to use a 74393 dual 4 bit counter driven by the 120 Hz signal C14. Also required was a spare 7404 inverter. This circuit provides a delay of approximately two seconds before reset occurs. By changing the C14 input to either C13 or C15 the delay can be changed to one second or four seconds respectively. An added bonus is that this circuit also provides power on reset.



DON VANSYCKEL, MIDDLEBURY, VERMONT

Concerning the article "HOW TO CHANGE DISK BUFFER LOCATIONS" by Tim Walkenhorst in Aardvark Journal Vol.1 No.6, it would be advisable to point out to your readers that the end of RAM pointer which BASIC uses must be changed. Otherwise, the disk buffer will over write BASIC strings and vise versa. This may be done by:

```
10 POKE(8960,PEEK(8960)-13
.
.
.
99POKE8960,PEEK(8960)+13:END
```

Note, if buffer # 7 is used the 13 in the example above must be 2

The POKEd value may also be done directly.

Second, if the current values of the #6 buffer END LO & HI are examined they are found to be the same as #7 buffer START LO & HI. Since #6 buffer does not write in #7 buffer it is obvious that the END LO & HI is the address of the end of the buffer +1, a fairly common trick for OSI (REF. OS-65D V3.0 USER'S MANUAL PRELIMINARY COPY OCTOBER 1978, P.58). With this in mind, the table locks out one byte of RAM more than the disk buffers require. This one byte actually costs a whole page (256 bytes) since BASIC's end of RAM pointer is only page specified. A solution to this is to add 1 to the addresses given in the table in the article making them:

TABLE OF VALUES FOR CHANGING DISK BUFFER LOCATIONS (5-1/4" DISK)

#6 BUFFER

FUNCTION	ADDRESS	32K	24K	CURRENT
START LO	17190 $4326	0 $00	0 $00	126 $7E
START HI	17191 $4327	120 $78	88 $58	50 $32
END LO	17192 $4328	0 $00	0 $00	126 $7E
END HI	17193 $4329	128 $80	96 $60	58 $3A
INPUT LO	17324 $43AC	0 $00	0 $00	126 $7E
INPUT HI	17325 $43AD	120 $78	88 $58	50 $32
OUTPUT LO	17347 $43C3	0 $00	0 $00	126 $7E
OUTPUT HI	17348 $43C4	120 $78	88 $58	50 $32

#7 BUFFER

FUNCTION	ADDRESS	32K	24K	CURRENT
START LO	17198 $432E	0 $00	0 $00	126 $7E
START HI	17199 $432F	112 $70	80 $50	58 $3A
END LO	17200 $4330	0 $00	0 $00	126 $7E
END HI	17201 $4331	120 $78	88 $58	66 $42
INPUT LO	17405 $43FD	0 $00	0 $00	126 $7E
INPUT HI	17406 $43FE	112 $70	80 $50	58 $3A
OUTPUT LO	17430 $4416	0 $00	0 $00	126 $7E
OUTPUT HI	17431 $4417	112 $70	80 $50	58 $3A

The BASIC end of RAM pointer may now be modified by:

```
10 POKE8960,PEEK(8960)-12
.
.
.
99 POKE8960,PEEK(8960)+12
```

Note, if buffer #7 is used the 12 in the example above must be 24

In addition it should be noted also that 8 inch disk uses 3K (12 page) buffers and the values in the tables should be adjusted accordingly.

(I haven't had time to look for the page count which will allow the use of 13 page buffer. Possibly someone else has investigated this.

KERRY LOURASH, DECATUR, ILLINOIS

I would like to point out an error in line 20 of the C2P BASIC Load program. The last data element should be 162, not 62.

I'd also like to remind you of the error in the E.Z. Lister program. Location 023A in the program should contain 9A, not 94.

You said you wanted to know how people liked the Journal's new format: I'm easy to please - as long as the listings are legible I'm satisfied. I hope more people are renewing their subscriptions.

ALBERT J. McCANN JR, LITCHFIELD PARK, ARIZONA

A short review of Mittendorf Engineering's Hi-Res Graphics Board and other related thoughts.

I finally have Mittendorf's High Resolution Graphics Board up and running. I bought the bare board version and scrounged up the parts I needed quickly and cheaply, except for the 40 pin jumper.

To connect the Hi-Res board you need the 40 pin jumper and a 16 pin jumper. The 40 pin jumper is the expansion buss and the 16 pin jumper brings in the needed timing signals. My video on the C1P refused to operate correctly because the 16 pin jumper acted like a 16 pin capacitor. I had to separate the wires of the jumper to cure this.

One other problem I had was with the 8T28 chips that drive the data bus on the 40 pin expansion buss. Mittendorf suggested removing them and putting jumper wires in the sockets to cure this. DO to DO etc. This fixed all problems.

The assembly manual was pretty good and I made no wiring or soldering errors.

The day before I installed my Hi-Res board, I received from Mittendorf Eng., some software for the board. It would not operate correctly with the Aardvark C1S ROM. I built a monitor ROM switch which I will explain later. PEEKing and POKEing Hi-Res memory worked OK with the C1S ROM. The Mittendorf software is a M.L. patch which adds new commands: LOAD CLEAR - CLEAR HI-RES MEMORY, LOAD NEW - OLD LOAD COMMAND, LOAD X,Y, 1 OR 0 - A PLOT X,Y, 1 OR 0 command with 1 lighting the dot and 0 turning it off.

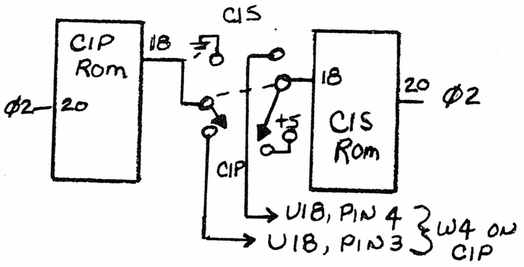
A sample 3D program is included which gives fantastic graphics. It takes about 5-7 minutes to build this display.

With the plot routine the 0,0 origin is the lower left corner, and 256,256 the upper right.

It is very easy to do 2D plots with this also.

The video of the Hi-Res board is combined with the C1P video so that the two graphics types can be used simultaneously. The C1P video can supply the numbers and base line and the Hi-Res video the plot. Hmmm! Scenery for an Adventure game!

Back to the C1S ROM problem. The plot routine uses the I/O in the old ROM and I haven't changed the jumps yet, so I made this simple adapter that switches between the C1P ROM and the C1S ROM:



Wire all the pins in parallel with each ROM except for Pins 18 and 20. Both phase 2 clocks are at W6. Switch shown in old C1P ROM position.

BARE BOARD $ 30.00
WHOLE KIT $185.00

PRINTING GRAPHICS WITH OS65D

Normally we are limited to ASCII characters when using 65D because non-ASCII is masked out to keep us from accidently sending control codes to other equipment (printers). That is not handy when you are programming games. Fortunately, it is easy to fix. Just POKE the offending code out of existance.

```
100FORX=9657TO9664: POKEX,234 : NEXT
110POKE9033,234:POKE9034,234
```

Then try "?CHR$(244)" and see the tank appear.

FOR SALE

OSI SUPERBOARD IN CASE WITH BUILT-IN 300-600 BAUD, REAL TIME CLOCK AND REVERSE VIDEO.
8K INCLUDES UNASSEMBLED 8K AARDVARK EXPANSION WITH MAJORITY OF PARTS. 3 AMP POWER SUPPLY, TAPERECORDER. SONY 12" MONITOR MODEL IS TELETYPE WITH INTERFACE (OPERATIONAL), PAPER, BOOK, AND ACCESSORIES, APPROX. $200 WORTH OF SOFTWARE- TOTAL PRICE $799.00.

SEND INQUIRES TO:

ROBERT CORWIN, RD#1, ROUTE 305 NORTH,
PORTVILLE, NY 14770

OR CALL: (716) 933-6749

WHATS NEW AT AARDVARK

It has, as usual, been an extremely busy couple of months since we published the last Journal.

The most fun item we have picked up in the month has got to be Labyrinth. It's a real-time monster hunt through a maze similar to that in Minos. You actually see a 3-dimensional maze as you walk through stalking monsters. The game runs in real time and you see real monsters moving across the corridors in front of you as you stalk them. I admit to having spent several hours debugging and testing this program. There weren't any bugs in it, but the testing was a lot of fun. In my personal evaluation it's one of the neatest games we've published in several months.
$13.95 on Tape and $15.95 on Disk.

We also have exciting news for those who like to experiment with Machine Code but don't really want to write any. AARDVARK is now marketing a Compiler. It's a very small, very tiny and very limited, but it's also very cheap. This Compiler takes an extremely limited set of Basic keywords and writes them into a Machine Code program. The Compiler itself is actually a program which sits in memory numbered from 8000 on up and which then PEEKs a program written between lines 0 and 7999 and converts it to a Machine Code program which places it at the top of memory. It has very few keywords so far. It has +, -, =, POKE, PEEK, IF, THEN, GOSUB and GOTO. By the time you get this Journal we should have had added the additions on to it for FOR, NEXT, USR(X). With a little ingenuity you can write video games with only the material in the Compiler. I spent the first couple hours after I received it writing a POTATO CHIP INVASION type program in Machine Code. It took me about two hours to write it, and it operated at Machine Code speeds.

As I have a personal enthusiasm for Compilers and have always wanted to have one, we are going to make some special deals on this Compiler in the hopes of getting a lot of people interested in it. The Compiler is available for all OSI systems for $14.95. That is incredibly cheap. Furthermore, if anyone makes an addition to the Compiler which we can then pass on to other users, then will receive an immediate refund for $14.95 and receive an AARDVARK Gift Certificate for between $25.00 - 100.00 depending upon the significance of the addition to the Compiler. The basic idea is to get a lot of people interested in and working on a Compiler.

We've also taken off in a new direction at Aardvark and have come up with a series of programs which I consider to be outstanding in their field. AARDVARK is finally in business. I've had a personal complaint for a long time that almost all business software for small computers was designed to be used by General Motors. It's always double entry, highly confusing and difficult to use. It normally requires a Bookkeeper or Accountant to set it up properly. It also seems to always be designed to run on at least a C2OEM or C3C.

Well now I'm finally getting around to doing something about it. Aardvark now publishes a series of programs designed for the Small Businessman. It was designed with the guy in mind who has 4 to 15 employees, who grosses less then 10 million dollars a year and doesn't want to have a full time accountant on his staff. This stuff is extremely easy to use, extremely clear and we made it very very, cheap.

The program cluster consists of a General Ledger program which records all of the transactions, does a Cash account balancing, a Profit-Loss balance and provides information for a Balance Sheet. The General Ledger program also has a neat little depreciation package which would be a good program by itself. You enter in the equipment, the date you got it, its value and its life span and the computer automatically figures the depreciation, stores it in a file, and enters it in to the Balance and Profit-Loss Sheets when needed.

We also have a Payroll Module that goes along with the General Ledger, an Inventory program, and an Accounts Receivable program. Except for the Accounts Receivable program they are all interactive. Thats means that one program will feed another program information without hand transferring it. Now the neatest thing is that we are going to sell the entire package of a Word Processor, General Ledger, Inventory, Payroll and Accounts Receivable for $299.95. That's got to qualify as one of the great give-a-ways in the Computer World. Maybe nobody will believe how good they are, because they are very, very cheap.

We also now have a all Machine Code Word Processor for the BASIC in ROM machines. We are currently running it on C1P and sometime in the next couple of days we hope to get it set up on a C4P and C1P with the C1E and C1S ROMs. I'ts a fairly simple little word processor, but it has most or all of the features that you need for low volume word processing. We are going to sell it for the rather unbelievable low price of $14.95. We haven't settled on a name for it as I go to press here but we'll probably call it something exotic like the Word Processor in Machine Code.

```
 60 REM FLIGHT OF THE MILLENIUM FALCON
 70 REM BY JOHN WILSON
 80 VI=540:IFPEEK(57088)>127THENVI=600
 90 POKE57000,2:SP=57089:POKESP,0:F1=0:G
OSUB1000
 100 PRINTTAB(5)"LONG AGO":PRINT:PRINT"I
N A GALAXY FAR AWAY...
 110 FORX=1TO32:PRINT:FORY=1TO65:NEXTY,X
:GOTO680
 120 M=0:W=32:D=2:SH=232:K=57100:MS=5503
9:S=53903:L=64:CH=300
 130 C=53760:C$="SHIP DAMAGED":TU=0:N=0:
K1=3:K2=5:K3=1:BL=32
 140 O=1:TH=3:ME(1)=111:ME(2)=79:ME(3)=2
26
 150 IFVI=540THEN300
 160 LC=32:W=25:MS=54115:S=53551:C=53478
:K1=252:K2=250:K3=254
 300 POKES,SH:FORX=OTOD:POKEMS+W*RND(O),
ME(RND(O)*TH+O):NEXT
 310 POKES,BL:P=PEEK(K):IFP=K2THENS=S-O
 320 IFP=K1THENS=S+O
 330 PRINT:PRINT:IFPEEK(S)<>BLTHEN360
 340 SC=SC+O:IFSC=CHTHEND=TH
 350 GOTO300
 360 IFNTHEN500
 370 TU=TU+O:IFTU>THTHENN=O
 380 POKESP,250:FORX=1TOLEN(C$):POKEX+C,
ASC(MID$(C$,X,1)):NEXT
 390 POKESP,0:GOTO300
 500 Y=S-3*L:POKES,32:FORX=1TO6
 510 POKEY+RND(8)*6+INT((RND(8)*6))*L,12
3+RND(8)*30
 520 POKESP,100+RND(8)*100:NEXT:POKESP,0
 530 FORX=1TO999:NEXT:GOSUB1000
 540 PRINT"YOUR SHIP IS DESTROYED":FORX=
1TO999:NEXT
 550 GOSUB1000:PRINT"THE EMPIRE HAS WON!
!"
 560 IFF1THENPRINT:PRINT:PRINT"       AGA
IN!!"
 570 FORX=1TO32:PRINT
 580 FORY=1TO80:NEXT:NEXT:PRINT:PRINT"SC
ORE: ";SC*10
 590 PRINT:PRINT:INPUT"TRY AGAIN";A$
 600 IFASC(A$)=89THENTU=0:SC=0:N=0:F1=1:
GOSUB1000:GOTO120
 610 GOSUB1000:PRINT"MAY THE FORCE BE WI
TH YOU"
 620 IFSC<300THENPRINT:PRINT:PRINTTAB(7)
"YOU NEED IT"
 630 FORX=1TO32:PRINT:FORY=1TO80:NEXT:NE
XT:POKE57000,1:END
 680 PRINTTAB(6)"THE FLIGHT
 690 PRINT"OF THE MILLENIUM FALCON":PRIN
T:PRINT
 700 PRINT"YOU ARE PILOTING THE
 710 PRINT:PRINT"MILLENIUM FALCON THROUG
H
 720 PRINT:PRINT"AN ASTEROID FIELD TRYIN
G
 730 PRINT:PRINT"TO ESCAPE THE EMPIRE
 740 PRINT:PRINT"FORCES.THIS IS YOUR ONL
Y
 750 PRINT:PRINT"MEANS OF ESCAPE.SEE HOW
 755 PRINT:PRINT"LONG YOU CAN LAST!!
 760 PRINT:PRINT:PRINT"LEFT SHIFT - LEFT
 770 PRINT:PRINT"RIGHT SHIFT - RIGHT
 780 PRINT:PRINT:PRINT"HIT SHIFT KEY TO
START"
 790 I=PEEK(57100):X=RND(8):IFI=1THENGOT
O790
 800 GOSUB1000:GOTO120
 1000 IFVI=600THEN1030
 1005 REM CHANGE SCREEN CLEARS FOR DISK
SYSTEM. SEE LAST YEARS
 1006 REM ARTICLE ON FAST SCREEN CLEARS.
 1010 A=PEEK(129):B=PEEK(130):POKE129,19
2:POKE130,215:S$=" "
 1020 FORS=1TO62:S$=S$+" ":NEXT:POKE129,
A:POKE130,B:RETURN
 1030 A=PEEK(129):B=PEEK(130):POKE129,0:
POKE130,212:S$=" "
 1040 FORX=1TO7:S$=S$+" ":NEXT:POKE129,A
:POKE130,B:RETURN
```

```
 4 REM WRITTEN BY JOHN WILSON
 5 REM FOR C1,2,4P   5/1981
 6 REM A TWO PERSON GAME
 10 VB=540:IFPEEK(57088)>128THENVB=600
 20 GOSUB1000:POKE57000,4:PRINT"ANTI-AIR
CRAFT ARTILLERY":PRINT:PRINT
 30 PRINT"DIRECTIONS ARE SIMPLE-":PRINT:
PRINT
 40 PRINT"LEFT GUN USE KEYS 1,2,3":PRINT
"RIGHT GUN USE N,M,<":PRINT
 50 PRINT"KEY 7 STOPS THE GAME.":PRINT
 60 PRINT"THE HIGHER THE PLANES":PRINT"A
LTITUDE THE GREATER THE
 70 PRINT"SCORE.":PRINT:PRINT"FIRST PLAY
ER TO REACH"
 80 PRINT"100 POINTS WINS.":PRINT:INPUT"
READY TO START";S$
 90 CB=8:GOSUB1000:SP=57089:POKESP,0:OF=
4096:POKE57000,6
 100 CO=53376:R1=239:L1=237:W=32:LC=64:K
B=57088:P1=4:P2=128:B=32:TA=6
 110 O=1:BO=46:TW=2:FORX=0TO6:READK(X):N
EXT:IFVB=540THEN150
 120 CO=53284:LC=32:W=25:P1=251:P2=127:F
ORX=0TO6:READK(X):NEXT:TA=1
 150 BL=CO+25*LC:FORX=OTOW:POKEX+BL,135:
NEXT:LE=CO+2*LC:RC=LE+W
 160 IFVB=540THENFORX=0TO5*LC:POKEBL+OF+
X,4:NEXT
 170 X=(BL-LC)+INT(W/2):RG=X+8:LG=X-9:PO
KERG,243:POKELG,246:LD=3*LC
 180 GOSUB510:GOSUB560:POKERP,R1:POKELP,
L1:M1=-1-LC:M2=1-LC
 190 POKE530,1:CR=0:CL=0
 200 POKEKB,P1:P=PEEK(KB):IFF1THENGOSUB4
00:GOTO230
 210 FORX=1TO10:NEXT
 220 IFP=K(0)THENF1=O:FI=RG+M1:RM=M1*3:G
OSUB600:RB=RG+M1:GOTO250
 230 IFP=K(1)THENPOKERG,243:M1=-1-LC
 240 IFP=K(2)THENPOKERG,242:M1=-LC
 250 POKEKB,P2:P=PEEK(KB):IFF2THENGOSUB4
50:GOTO280
 260 FORX=1TO8:NEXT:IFP=K(6)THEN840
 270 IFP=K(3)THENF2=1:FI=LG+M2:LM=M2*3:G
OSUB600:LB=LG+M2:GOTO300
 280 IFP=K(4)THENPOKELG,245:M2=-LC
 290 IFP=K(5)THENPOKELG,246:M2=1-LC
 300 POKERP,B:RP=RP-RF:POKERP,R1:CR=CR+R
F:IFCR>WTHENGOSUB500
```

```
 320 POKELP,B:LP=LP+LF:POKELP,L1:CL=CL+L
F:IFCL>WTHENGOSUB550
 390 GOTO200
 400 POKERB,B:RB=RB+RM:IFRB<COTHENF1=ZE:
RETURN
 410 Q=PEEK(RB):Q1=PEEK(RB+O)
 420 IFQ<>BORQ1<>BTHENGOSUB650:F1=ZE:RET
URN
 430 POKERB,BO:RETURN
 450 POKELB,B:LB=LB+LM:IFLB<COTHENF2=ZE:
RETURN
 460 Q=PEEK(LB):Q1=PEEK(LB+O)
 470 IFQ<>BORQ1<>BTHENGOSUB670:F2=ZE:RET
URN
 480 POKELB,BO:RETURN
 499 REM BLANKS PLANE CHOOSES PLANES ALT
.&SPEED
 500 POKERP,B:CR=ZE
 510 RL=INT(RND(0)*5):IFRL=LLTHEN510
 515 RF=0:IFRL>0THENRF=TW
 520 RP=RC+RL*LD:RETURN
 550 POKELP,B:CL=ZE
 560 LL=INT(RND(0)*5):IFLL=RLTHEN560
 565 LF=0:IFLL>0THENLF=TW
 570 LP=LE+LL*LD:RETURN
 600 POKEFI+OF,2:FORX=224TO226:POKEFI,X:
POKESP,X:NEXT
 610 POKEFI,32:POKEFI+OF,8:POKESP,0:RETU
RN
 650 IFQ=R1ORQ1=R1THENRS=RS+10-RL:GOTO73
0
 660 RS=RS+10-LL:GOTO700
 670 IFQ=R1ORQ1=R1THENLS=LS+10-RL:GOTO73
0
 680 LS=LS+10-LL:GOTO700
 700 POKELP+OF,TW:FORX=25TO32:POKELP,X:P
OKESP,200-X*TW:NEXT
 710 POKESP,ZE:POKELP+OF,CB:GOSUB550:GOT
O750
 730 POKERP+OF,TW:FORX=25TO32:POKERP,X:P
OKESP,125+X*TW:NEXT
 740 POKESP,ZE:POKERP+OF,CB:GOSUB500
 750 IFRS>99ORLS>99THEN800
 760 PRINTCHR$(13)TAB(TA)" "LS"
 "RS"    ";
 770 RETURN
 800 FORX=1TO1000:NEXT:GOSUB1000:IFRS>LS
THENS$="RIGHT":GOTO820
 810 S$="LEFT"
 820 PRINT:FORX=1TO25:PRINTTAB(X)"THE "S
$" GUNMAN WON!":NEXT
 830 INPUT"TRY AGAIN";S$:IFASC(S$)=89THE
NRS=0:LS=0:GOSUB1000:GOTO150
 840 POKE57000,1:POKE530,0:GOSUB1000:END
 900 DATA4,8,2,64,128,32,2,251,247,253,1
91,127,223,253
 1000 IFVB=600THEN1040
 1010 X=PEEK(129):Y=PEEK(130):POKE129,19
2:POKE130,215:S$=" "
 1020 FORI=1TO62:S$=S$+" ":NEXT:POKE129,
255:POKE130,231:R$=CHR$(CB)
 1030 S$=R$:FORI=1TO62:S$=S$+R$:NEXT:POK
E129,X:POKE130,Y:RETURN
 1040 X=PEEK(129):Y=PEEK(130):POKE129,0:
POKE130,212:S$=" ":FORI=1TO7
 1050 S$=S$+S$+" ":NEXT:POKE129,X:POKE13
0,Y:RETURN
```

AARDVARK TECHNICAL SERVICES, LTD.
2352 South Commerce
Walled Lake, MI   48088
(313) 669-3110

BULK RATE
U. S. POSTAGE
PAID
WALLED LK.,MI.
Permit No. 5

J. COOK
402 N. SHORE DR.
BELLINGHAM
WA, 98226